Shanghai Jiao Tong University

Department of computer science and engineering

统计学习理论与方法期末作业

Author: 陈欣祺

An Assignment submitted for the SJTU:

统计学习理论算法 *CS7304*

2021 年 12 月 29 日

目录

1 数据预处理

对于题目所给的训练集和测试集要先做数据预处理工作，其中测试集是只有数据（.npy 格式）没有标签的，而训练集既有数据也有标签（label_train.csv 文件）。由于给出的实验数据即使特征数据，所以无需做归一化等其余预处理操作。数据预处理主要分为载入数据和数据集划分两部分，下面将依次介绍。

1.1 载入数据

实验数据预处理的第一步是载入数据，在 *train* 文件夹下依次读取数据，并拼接成一个整体。每一个 *.npy* 文件的大小为 $100 * 15$，其中 100 为时间序列，15 为 *channel*，为了方便后续的深度学习网络训练，这里将每一个 *numpy* 数组取转置，让 *channel* 在 *shape* 的第二个位置。训练数据的总大小为 $(1334 * 15 * 100)$，利用同样的处理方法，得到测试集数据大小为 $(348 * 15 * 100)$。在依次拼接数据的同时，查找 *label_train.csv* 文件对应的标签，再得到数据相应的标签，数据集共有 20 个类别，分别用 0,1,2...19 来表示。至此完成数据的载入工作，以上所有操作均可在项目的 *utils.py* 文件中找到。

1.2 数据集划分

实验数据预处理的第二步是数据集划分，主要分为两种情况来进行。对于深度学习模型，本次实验中选用 pytorch 搭建网络，数据集需要先读取成 *torch.utils.data.Dataset* 形式的数据集，再制作成 *torch.utils.data.DataLoader* 的格式。且需要提前将训练数据划分为训练集和验证集，方便训练时选择在验证集上表现优异的模型，这里利用 *sklearn.model_selection.train_test_split* 按照 8:2 的比例划分，并固定随机种子划分数据，便于实验复现。而对于机器学习算法来说，只需要数据原始的形式即可，但是在 sklearn 中的各种机器学习算法中，要求输入是二维的，所以要将后两个维度合为一维。对于训练集和验证集划分，这里选用十折交叉验证的方法，并且将数据打乱顺序。具体的预处理方法在 *main.py* 中可以找到。

2 选用模型

本次实验要求选取四种统计学习模型分别实验，我这里选用了课上讲述的三种机器学习监督模型：随机森林、支持向量机、K 近邻，和一种深度学习模型：MLP，其中机器学习模型用来作为 baseline，深度学习模型用来提升在测试集上的准确率，下面我将分别介绍这些模型。

2.1 随机森林

随机森林是一个机器学习中很常见的模型，在很多论文中被当做对比的 baseline。随机森林能够处理具有高维特征的输入样本，而且不需要降维，能够评估各个特征在分类问题上的重要性，并且在大数据集中有着较好的表现，所以我选择其为本次实验的模型之一。

这里使用 *sklearn.ensemble* 中的 $RandomForestClassifier$，随机森林的树个数，也就是 $n\_estimators$ 是一个超参数，在之前的工作中有学者提到，对于大多数任务该参数设为 100 左右表现较好 [1]。这里选用 100 个树，不改变其他参数进行训练。

2.2 支持向量机

支持向量机 (SVM) 在二分类中经常被用到，可以很好的解决感知机无法解决的线性不可分问题。其决策边界是对学习样本求解的最大边距超平面，样本中距离超平面最近的被称为支持向量。SVM 可以通过核方法进行非线性分类，是常见的核学习方法之一，其中较为常用的核函数有高斯核、径向基函数核、线性核等。SVM 同样可以推广到多分类中，可以分为两种方法：one-against-one 和 one-against-the-rest。其中 one-against-one 方法设训练集数据共 N 个类，在每两个类之间都构造一个 binary SVM，对于新数据应用投票策略，每个 binary SVM 根据其决策函数对新数据有一个预测，预测为相应类则分数加一，最终得分最高的即为被预测成的类别；one-against-the-rest 方法对于每一个类，将其作为 +1 类，而其余 N-1 个类的所有样本作为-1 类，构造一个 binary SVM。

模型选用 $sklearn$ 中的 $svm.SVC()$，核函数选用径向基函数，惩罚系数 $C = 3$，其余均选用默认参数。

2.3 K 近邻

K 近邻 (K- Nearest Neighbor, KNN) 是本课程最早讲到的几个算法之一，每个样本都可以用它最接近的 K 个邻近值来代表，近邻算法就是将数据集合中每一个记录进行分类的方法。在课堂中老师讲到，随着数据维度的增加，KNN 会有着维度灾难等问题，它不适合高维数据而更适合低维数据。针对大数据的分类问题，它存在着如下缺点: 1) 对每一个待分类的文本都要计算它到全体已知样本的距离，才能求得它的 K 个最近邻点，而大数据的典型特点就是数据信息海量、价值密度低，这就显然出现了很大的无效计算量; 2) 大数据出现新的类别的概率极大，而 KNN 算法的邻居都是已知的类别样本，也就导致了对新样本的无知或者误判。所以相比其余模型它可能更适合二分类任务，而不是像本次实验任务的二十分类，此处选用该模型是为了巩固课内学到的知识。KNN 模型选用 $sklearn.neighbors$ 中的 $KNeighborsClassifier$，主要参数 $n\_neighbors$ 需要调参尝试，最终实验证明当 $n\_neighbors = 10$ 时效果较好。

2.4 多层感知机

多层感知机 (Multi-Layer Perceptron, MLP) 是由感知机推广而来，可以解决线性不可分问题，最主要的特点是有多个神经元层，因此也叫深度神经网络。它的训练过程可以分为前向传播和反向传播两步，通过利用优化函数将训练集损失函数最小化，即可一步步得到表现效果更好的模型。由于本次实验数据 $channel = 15$, $feature = 100$，维数不是很高，所以无需过深的网络结构，防止造成过拟合结果。

在 pytorch 中 MLP 可由多个 $nn.Linear$ 层构成。这里我设置了三层网络，最后一层为 $flatten$ 层，输出维度为 20。此外还有 relu 激活函数、batchnorm 和 dropout，为了防止网络过拟合。MLP 的主要参数如下表 1 所示。

Layer	Parameters		
	Input	**Output**	**Single Paramter**
layer1	15	128	×
bn1	×	×	100
layer2	128	64	×
bn2	×	×	100
flatten	×	×	start_dim=1
layer3	6400	20	×
dropout	×	×	0.8

表 1: MLP 参数表

2.5 BERT

近些年来，随着 Transformer 结构在自然语言处理方面的迅速发展，越来越多的工作开始研究这一模型。Transformer 同样在时间序列处理中有着很好的表现 [2] [3]，于是对于本次实验我想尝试 Transformer 的 encoder 部分 (即 BERT 的主要部分) [4] 作为网络结构，将部分输入数据掩盖掉，即以一定的概率替换为 0，训练 encoder 可以学习到输入数据之间的相互联系，再额外利用一个 GRU 作为分类器，最终经过两次训练可以得到预测结果。该部分代码在 *model.py* 中，但还需要进一步完善，由于发现 MLP 的测试结果已经较好，这部分工作可以之后继续完成。

3 训练与模型评估选择

由于数据集划分方式不同，机器学习模型和深度学习模型训练的方法也不一样，下两节分别介绍训练流程，并分别介绍本次作业中采取的模型评估与选择思想。

3.1 机器学习模型训练与评估

对于三种机器学习模型，没有梯度下降等参数更新方法。所以设置 $epoch = 300$，对数据集不固定随机种子，每一个 epoch 进行随机划分，并应用十折交叉验证，选择其中在验证集上表现最好的一折。这里选用的评估思想便是交叉验证，每一轮应用不同的训练数据和验证数据，并找到效果最好的模型。

训练时三个模型都是通过 $model.fit(X_train, y_train)$ 的方法，再利用 $model.predict(X\_vali)$ 得到模型在验证集上的预测结果。

3.2 深度学习模型训练与评估

对于深度学习模型，需要确定的训练参数相较机器学习要多很多，包括训练轮数、学习率、损失函数等等。MLP 模型的训练超参数如下表 2 所示。

在训练时要将模型设置为训练状态 $model.train()$，而在验证和测试时，由于 MLP 中有 BatchNorm 和 Dropout 方法，故需要将模型设置为验证状态 $model.eval()$ 保证二者的参数不变，才能得到真正的结果。

Hyper parameters	Value
初始学习率	1e-4
学习率衰减	每 50 个 epoch 乘 0.95
epoch	1000
batch size	128
损失函数	CrossEntropy
优化函数	Adam(weight_decay=0.001)

表 2: MLP 训练设置超参数表

4 实验结果

4.1 随机森林结果

随机森林在训练集和验证集上表现最好的结果为：Epoch 173/300 第 5 折交叉验证，训练集 Acc: 1.000000, 验证集 Acc: 0.947368。

随机森林在测试集上的结果如下图所示。

图 1: RF 结果

4.2 支持向量机结果

支持向量机在训练集和验证集上表现最好的结果为：Epoch 11/300 第 1 折交叉验证，训练集 Acc: 0.998333, 验证集 Acc: 0.955224。

支持向量机在测试集上的结果如下图所示。

图 2: SVM 结果

4.3 K 近邻结果

K 近邻在训练集和验证集上表现最好的结果为：Epoch 188/300 第 5 折交叉验证，训练集 Acc: 0.834305, 验证集 Acc: 0.902256。

K 近邻在测试集上的结果如下图所示。

图 3: KNN 结果

4.4 多层感知机结果

MLP 在训练集和验证集上表现最好的结果为：Epoch 845/1000 Best Train Loss 0.0328 Acc: 0.9897. Validate Loss 1.3524 Acc: 0.9064。

MLP 在测试集上的结果如下图所示。

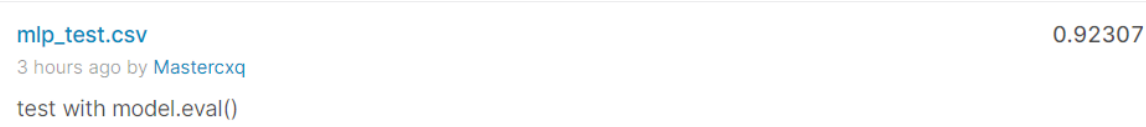

图 4: MLP 结果

5 分析与总结

由于 MLP 方法的测试集占比 20%，大于机器学习模型中的 10%，故训练时测试集的表现要稍弱于机器学习模型，但是在测试集上的表现效果更好。从以上实验结果可见，四种模型在测试集上的表现效果 MLP 最好，SVM 第二，随机森林第三，K 近邻第四，与我们对模型分析时的预期大体一致。如果换用 BERT+classifier 的方法应该会达到更好的表现效果，这些模型的表现结果都是与他们本身的特性有关的，在时间序列处理上，深度学习模型的表现能力要强于大部分的机器学习模型，但是深度学习的成功是以更大量的数据和良好的数据预处理过程为基础的。

通过本次大作业，让我对课内学习到的机器学习算法有了更深的认识，并且增强了我的代码工程能力，使我受益匪浅。

参考文献

[1] Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How many trees in a random forest? In *International workshop on machine learning and data mining in pattern recognition*, pages 154–168. Springer, 2012.

[2] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[3] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, WEI Ying, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. 2020.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.